AMATH 482/582: HOMEWORK 5

KATIE HIPPE

AMATH Department, University of Washington, Seattle, WA
kahippe@uw.edu

Abstract. The SEVIR dataset is a collection of satellite imagery, radar mosaics, and lightning detections from storm events. It is a classic dataset used in atmospheric science to display common machine learning techniques, including PCA, classification, and regression, as we may first compress each image, then evaluate on whether is has a thunderstorm (defined here as containing any lightning at all), and extend to how many lightning flashes are in it.

1. Introduction and Overview

One of the most important open questions in atmospheric science today is how best to predict the weather. While we have access to plenty of numerical predictors, like systems of differential equations, and laws governing physical systems, solving these equations can take a long time and requires initial conditions of measurements that can be hard or impossible to get a map on the correct scale for. Instead, we may now turn to machine learning techniques to create predictions on a much faster scale.

1.1. **Data.** However, much of the weather data available to the public is of too great a size and complexity to be used on personal computers. So, a smaller dataset was constructed: The Storm EVent ImagRy (SEVIR) dataset. SEVIR is a collection of weather events as captured by satellite and radar that are spatially and temporally aligned. These events, primarily thunderstorms, contain five major types of images. Find examples of the types of images in the SEVIR dataset in Figure 1:

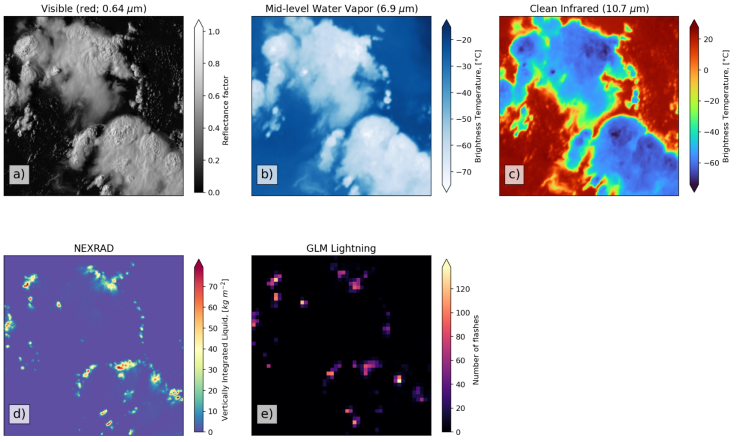


Figure 1. Examples of different data contained within the SEVIR dataset

Date: March 20, 2025.

These images are: visible satellite imagery, infrared satellite imagery (water vapor and clean), NEXRAD radar mosaic of vertically integrated liquid, and GLM flashes, which are detections of inner cloud and cloud to ground lightning events. Each set of data is represented as a sequence of gray scale images, except for the lightning, which instead contains information about the flash times and locations. Each storm event in the SEVIR dataset has four hours of data broken up into 5 minutes time samples. The number of lightning flashes per event is highly variable: for about 50% of the data, there are no lightning events, but when lightning does occur there can be as many as 6,000.

For training purposes, many additional images in this dataset contain random images of normal weather situations, not weather events, so that our classifiers see a variety of images. Much of this data comes from the National Weather Service (NWS). Their database contains additional connections to the NOAA's Storm Event Database, which has information of the type of weather, storm impacts, and a description of the event [1].

1.2. **Problem.** For the purposes of this paper, we will be focusing on the processes of PCA, classification, and regression. The SEVIR dataset, even a smaller version like the one we will be using here, contains a large amount of information due to the high spatial resolution of each image, as well as the fact that we have five separate sets of information to deal with. Thus we will use PCA to keep a majority of the information while significantly decreasing the dimensionality of the dataset. Next, we will use classification techniques to determine if, for a particular image, there is a thunderstorm. Finally, we will use regression to determine how many lightning flashes are in a particular image.

2. Theoretical Background

We first obtained 619,213 samples from the SEVIR dataset, containing 5 subdata sets, each with dimenionality of at least 192×192 and at most 768×768, depending on which sub-feature we are preoccupied with.

2.1. **Dimension reduction using PCA.** . Given the high resolution of these images, we wish to reduce the dimensionality significantly. To do so, we conducted a Principal Component Analysis (PCA) to find the most important components of our data. To do so, we used Singular Value Decomposition (SVD) (1):

$$X = U\Sigma V^T \tag{1}$$

where X is our training data matrix. It has been decomposed into three component matrices: U, the left singular vectors, V, the right singular vectors, and Σ, a diagonal matrix filled with the singular values of X sorted in descending order.

Once we have found U using the entirety of our training data, we may use it to project a given data sample z (all 784 pixels) onto the determined PC modes:

$$z = U^T z$$

where we have as many PC modes as pixels in our initial data. However, in the case of our data this means 784 PC modes, and we wished to reduce the dimensions of our data. So instead we may project on $k-$PC modes, so instead of 784 modes we only have k, using (2)

$$z_k = U_k^T z \tag{2}$$

where U_k^T has the first k column vectors kept intact while the column vectors from $k+1$ and onward are set to 0. When we apply this to a data sample z, the result will be a reduced-dimension in $k-$PCA space, which we may compare with other data in the same $k-$PCA space.

2.2. **Classification and Regression using Machine Learning.** To classify our data, we first used a Ridge Regression classifier (3). This model performs a linear regression including a regularization parameter.

$$\beta = \text{argmin}_{\beta \in \mathbb{R}^J} \frac{1}{2\sigma^2} ||X\beta - Y||^2 + \frac{\lambda}{2} ||\beta||_2^2 \tag{3}$$

In this case, we are conducting a more general polynomial regression to find the parameter vector β that best transforms the data matrix X into the label matrix Y, that is, minimize the error over all β possibilities. The regularization parameter $\frac{\lambda}{2}||\beta||_2^2$ is what specifically defines this process as Ridge Regression. This parameter helps when X is nearly linearly dependent. The specific value of λ is determined via cross-validation using MSE (see 3 for more detail).

This type of cross-validation involves sub-setting our data into discrete units, and testing our model run on each of them. Thus we will get a unique accuracy for each subset, and may take the average and standard deviation so we may have a better degree of confidence in our accuracy score. This will also help us determine whether the model is under- or over-fitting, as a high standard deviation would imply the model works better for certain data than others and could be over-fitted.

Using this classifier, we may evaluate our data to find a solution to both our goals. First, we want to classify the image into two categories: thunderstorm, and no thunderstorm. We may do this by determining whether there exists any lightning flashes in a given image, as a simple binary sort. Then, we wish to fit a regression (output a continuous list of parameters) to determine the number of lightning flashes in a given image.

In addition, we used a Linear Regressor and Decision Tree Regressor. Though we did not explicitly cover them in class, they are similar methods: the Linear Regressor is the same as Ridge Regression but without the regularizer term. The Decision Tree is a bit different, as it breaks down the dataset into increasingly smaller subsets while creating an associated decision tree.

2.3. **Neural Networks.** If these supervised machine learning techniques don't work as well as we want them to, we may turn to FCNs, a Neural Network. Neural Networks connect our initial high dimensional data to a much lower dimension set of classifications, with one or more hidden layers of different dimensionality to facilitate classification. The input layer has our original dimension (here 39) whereas the output layer will have a lowered (here 1 or 2) to represent the potential classifications.

In our Neural Net, we connect the neurons through various hidden layers. The unit is a weighted sum of its inputs, and thus the connections between neurons are a simple linear function. This sum goes through a nonlinear function to get to the next layer, which often has a different dimensions. In this case, our activation is ReLu 4, which zeros out all negative input.

$$f(x) = \max(x, 0) \tag{4}$$

As we step through the different epochs (each time looking at our data), the learning algorithm updates the weights and biases of each node until we arrive at an optimized training accuracy. At each epoch, we calculate the accuracy on our validation set, as well as compute the training loss 5, which is the error when the model makes predictions on the training data. This loss informs back-propagation to fine-tune the weights. As the model improves its classifications, this training loss should drop.

$$L(\hat{y}, y) \propto ylog(\hat{y}) + (1 - y)log(1 - \hat{y}) \tag{5}$$

3. Algorithm Implementation and Development

To develop and run these functions and algorithms, we used Python 3.12.8 in Visual Studios Code. For general numerical methods, we used NumPy [3]. For plotting, we used Matplotlib [4] and Seaborn [6]. For machine learning methods like Ridge Regression, Linear Regression, and Decision Trees, we used scikit-learn [5]

First, we examined our dataset to attempt to use PCA on it to reduce the dimensionality. Once we had our data in a reduced form, we sorted into training, testing, and validation sets. Then, we fit a linear regression, in this case Ridge Regression. This classifier internally runs different values of λ and determines the best one for the particular dataset by minimizing MSE:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N} |f(\hat{X})_i - Y_i|^2$$

where $f(\hat{X})_i$ is the predicted value from our current model $\hat{f}$ and Y_i is the actual value, and N is the number of samples we have. By minimizing MSE, we optimize our λ from a predetermined options list. Thus function also automatically conducts a cross-validation, with which we can determine a better idea for how well our model is working.

After we are satisfied with the predicting skill of the model, we used it first to classify whether an image contained a thunderstorm, and then to regress to find the number of lightning flashes. We then compared these against simple Neural Networks to see if we could improve our performance.

4. Computational Results

4.1. **Data processing and dimension reduction.** Our first computational result should come from the results of our PCA analysis. However, once the dataset was downloaded, we encountered significant setback in the context of combining datasets and isolating the correct features from each dataset. Additionally, the raw dataset was too large to be housed on a personal computer, so after some trial and error, performing PCA analysis was scrapped in favor of a different method of cleaning the data [2].

In the context of thunderstorms, we may apply previous knowledge to inform how we reduce the dimension of the data. A more intense storm is often associated with more lightning, so we may derive proxies for estimating storm strength from each data set. From the visible channel, we may use the magnitude of reflectance. From the water vapor and clear infrared channel, we may use how bold the brightness temperatures are. Finally, we may use how much vertically integrated water there is. Thus from each type of data, we may extract various percentiles from each image and variable, in this case we're using 0, 1, 10, 25, 50, 75, 90, 99, and 100. Thus we have reduced our dimensionality all the way down to 9 for each image type.

Once we had all of our different datasets down to 9 features each, we combined the different images together to create our X data. We combined horizontally the clean infrared channel, the water vapor channel, the visible channel and the vertically integrated liquid. Since we concatenated in the same way each time we sampled from our dataset, we might test different numbers of features on our regression techniques by keeping the first n out of 36 total features from this reduced dataset.

Our final effort to process our data was to create the label vector for our two goals. The number of lightning flashes are summed across our time domain, and if there is a lightning flash in the last five minutes, the image is classified as containing a thunderstorm. For our regression, the image is given the sum of all lightning flashes in the past five minutes within the image.

4.2. **Classification using Ridge Regression.** Our first task involved classifying whether a an image contained a thunderstorm. To do so, we evaluated the truncated data as processed above against a binary label set of whether there was a lightning flash in the past five minutes.

First, we ran the Ridge Regression keeping only one feature from the clean infrared channel. This yielded a training accuracy of .7721, a testing accuracy of .8149, and after cross-validation, we determined the standard deviation of the training accuracy to be 0.0344.

Since we had up to 36 different features, we tested the training and testing validation as a function of the number of features we kept. These are graphed below in Figure 2.
We can see that the testing and training accuracy are very similar for a majority of the features, with some slight differences at 5 features and below. Both accuracies jump up at the very end, as

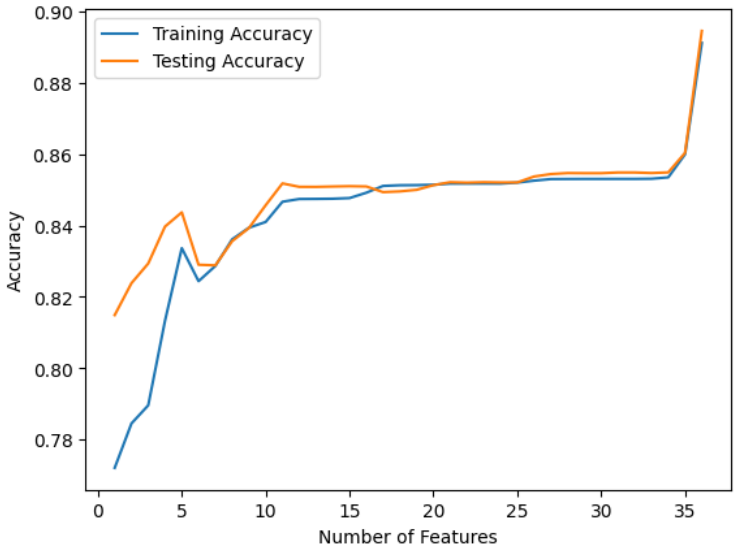


FIGURE 2. Testing and training accuracy based on number of features

we've increased to our maximum number of features. Since the accuracy lines are so similar, we can reliably say that the Ridge Regression model works well for our entire dataset without over- or under-fitting.

We have a surprisingly high baseline accuracy. We may consider this in the context of our dataset. About half of our images contain no lightning flashes at all, so our baseline accuracy should be around .5 anyway. However, we did reach nearly .9 accuracy after including only 36 features. This tells us that lightning has a very high correlation with very basic information from an image, in this case the very first feature, which corresponds to the 0^th percentile from the clean infrared channel. Thus lightning is very highly correlated with brightness temperature, which makes sense considering lightning is a flash of light.

4.3. **Regression using a variety of classifiers.** Next, since we'd seen that the Ridge Regression worked the best whilst using all of our available, we attempted an ensemble of regressors for our second task: identifying how many lightning strikes were in a given image. We began our analysis using only one feature, as we started before, and then progressed to all 36 features. For this step, we used the Ridge Regression as before, as well as a Linear Regressor and Decision Tree Regressor. To evaluate how well these regressors worked, we may compare the true number of flashes and the predicted number of flashes on the validation set in a one-to-one plot.

Since we have so many data points, we binned the plotted data and labeled them with a scaling colorbar based on what percent of the total dataset they represented. Thus we may evaluate how well our methods worked. The Ridge Regression did not work very well at all, as the predicted number of flashes was often much lower than the actual, and didn't follow a linear trend. The Linear Regression was slightly better, as we can see there does exist a correlation, but the slope is not very similar to the $y = x$ line.

Given that the linear regressors were not working very well on this particular dataset, we turned to different nonlinear methods. The Decision Tree regression technique, which involves splitting the data up into smaller and smaller subgroups and creating an associated decision tree, ended up performing the best of all, as it has the highest percent of data along a mostly-linear line.

Next, we upped the number of features in another attempt to improve the linearity. As before, we achieved the best results when we considered all 36 features together, show in Figure 4 below. We may compare this with Figure 3 above and notice that each of our regressors have improved

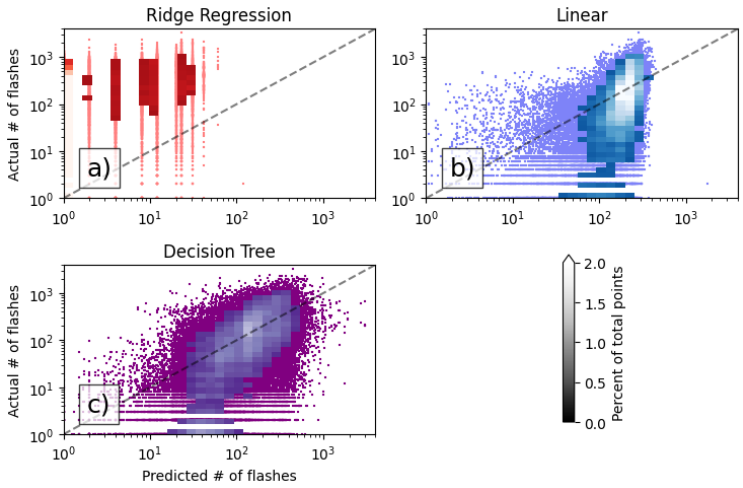


FIGURE 3. Various regressions for one feature

significantly, now matching the overall shape of a one-to-one function much better. Ridge Regressor still performs the worst, though now the Linear Regressor and Decision Tree have better shapes. However, both Linear and Decision Tree have a wide range of values that are not well-correlated.

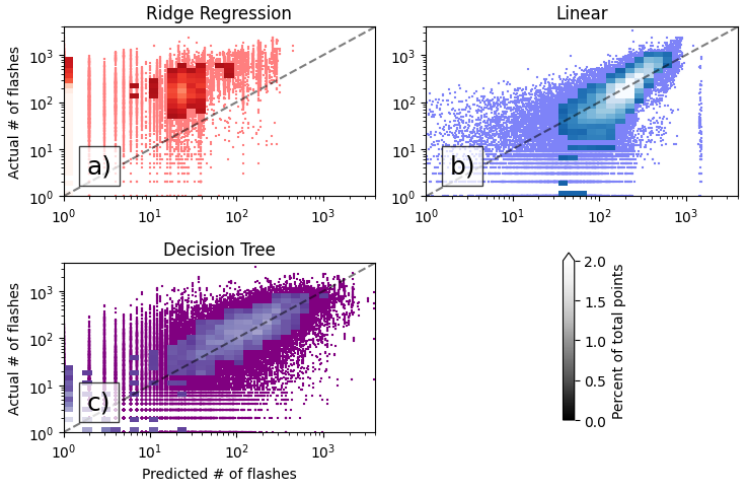
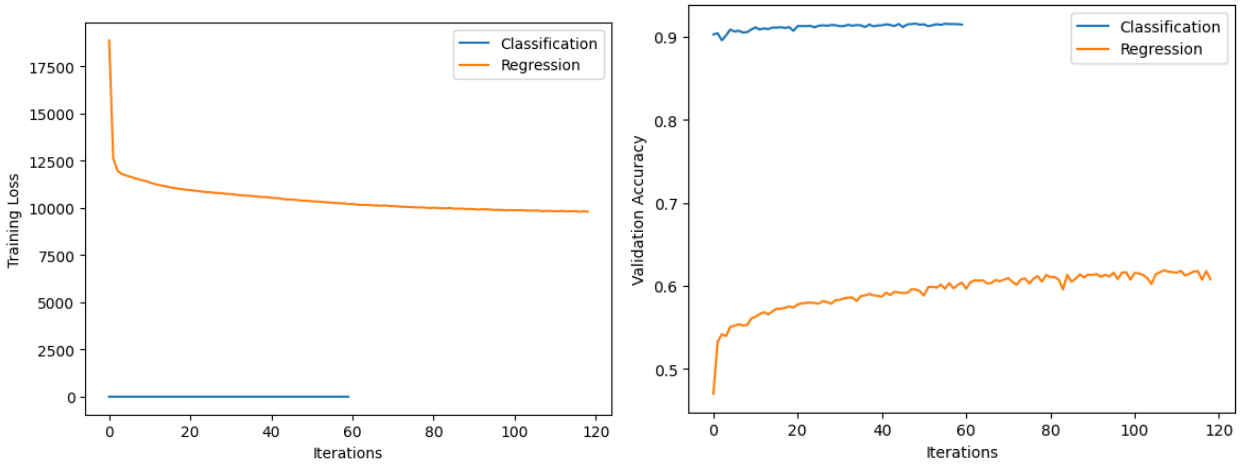


FIGURE 4. Various regressions for thirty-six features

Despite improving the overall shape of our graphs, we see that these models still do not predict the number of lightning flashes very accurately. This could be because of our high amount of bins. In previous problems in class, we used a finite bin number. Both the FashionMNIST and MNIST datasets had about 10 different classifications, while the number of lightning strikes is a continuous metric with a wide variety in range.

Additionally, we may note the differences in the Ridge Regression versus the Linear Regression. Linear and Ridge Regression are nearly identical, save for the regularization term present in Ridge Regression. The regularization term is meant to prevent overfitting, and thus the model should work better on testing and validation data than if we had just a linear term. However, the results of this analysis suggest otherwise, as the Linear Regression worked better than Ridge. This could be due to the relatively low number of features we had to work with, and since we did not conduct PCA, we don't know that these particular features are the optimal ones for our regressions.

4.4. **Regression and Classification using Neural Networks.** Our final effort to solve this problem was to use Neural Networks. Since our data was in a significantly different format than the FashionMNIST dataset, using the same type of Fully Connected Neural Network did not work. After encountering numerous errors, we turned to built-in networks from `Scikit-learn`, specifically `MLPClassifier` and `MLPRegressor`. While these are built-in functions, we could still change some of the hyperparameters, including the optimizer (set to Adam), the number of iterations (120), and a validation fraction (0.1) with which to track validation scores. We may use the training data and set aside a fraction of that for validation, then use the test data to see our final accuracy. We also turned on early stopping, which halted th iterations when a maximum was reached for validation accuracy. For both of these models, we used all 36 features of our dataset.



(A) Training loss (B) Validation accuracy

FIGURE 5. Result

For the classification, we got an accuracy of 0.9133 for our testing data, which is not much higher than the $\approx .9$ accuracy that the simple Ridge Regression yielded. The regression to count the number of lightning flashes also did not perform very well, as we only got to about a .6 accuracy for the testing data, and had a very high error (calculated as mean-squared error, see 3). Thus our attempts at regressing the data to count the number of lightning strikes was largely unsuccessful for both supervised and unsupervised machine learning techniques.

5. DISCUSSION

5.1. **Understanding the problem.** These data analyses contributed significantly to the understanding of both the problem and the dataset. Working through the PCA analysis, even though it didn't work, helped to learn more about the various features within the dataset. Our replacement analysis that used percentiles also helped support the baseline understanding of the problem, as now we know what features are most important when predicting lightning.

Through our regression techniques, we found that thunderstorms are much easier to predict than number of lightning flashes, as the same model yielded about 90% accuracy using only 36 features for thunderstorms but did not accurately predict the number of lightning flashes at all. This is most likely due to thunderstorms being a classification problem, while lightning flashes exist on a much larger, more continuous spectrum.

5.2. **Class method contribution.** The class methods were very helpful as they provided a framework within which to conduct our own analysis. The SEVIR dataset is a popular one, with plenty of other examples of machine learning techniques utilizing the information within it. While the

problem chosen is a common example of meteorological machine learning analysis, we used specific methods from the course to compare against existing literature. Given the better understanding of the frameworks within simpler methods, like Ridge Regression, we are better apt to expanding this understanding to other methods, in this case Decision Tree.

Over the course of the class, we utilized various methods for understanding the most important part of datasets. We have countless measurable data points, and while classical numerical methods thrive on having robust datasets for initial conditions to work off of, machine learning techniques are all about reducing the data we have towards a more manageable size. While we did not use PCA on this dataset, taking percentiles is still very much within the spirit of using only the most important information from a dataset.

We were able to run some of our data analysis techniques, namely regression, for a few different purposes, and ended with a reasonable model to predict both the existence of thunderstorms and how many lightning flashes a storm contains. Since we did a very involved model in Homework 4, we have a good enough understanding of Neural Networks such that we can turn to the built-in models instead and do some basic tuning to suit them to our needs.

5.3. **Possible extensions.** As discussed above, trying to run PCA on such a large dataset did not work, especially when constrained to a personal laptop. While we did find a pre-processed version of SEVIR online, it was severely limited to only 36 features, which is not as conducive to the kind of analysis that we have been doing recently. Especially in terms of neural nets, as we'd already accomplished such a high accuracy that running such a computationally expensive process was not necessary. A different extension could involve finding a different pre-processed SEVIR dataset that preserved more of the original data, or isolating just one of the four images to compare directly against the lightning data.

Additionally, since over half of the data does not contain lightning flashes, the regression techniques did not work as well as they could have. In this case, we could have used more robust nonlinear methods to better predict the number of lightning flashes. Since this is a continuous distribution instead of a discrete, it was also harder to generalize our regression techniques. To improve this part of the model, we may modify our FCN from Homework 4 to allow for continuous variables in the output, though attempts were unsuccessful so far, it is possible. We might also conduct more hyperparameter tuning on the built-in functions, as for the purposes of this paper we merely used the default parameters.

6. Summary and Conclusions

Over the course of this homework, we attempted dimension reduction and simple machine learning techniques to better understand the SEVIR dataset as well as how we may predict thunderstorms and lightning flashes. Since our regressors worked well to predict whether a thunderstorm was present in a given picture, we may use it in a more general sense to predict whether certain conditions are conducive to lightning and evaluate that risk. Machine learning is quickly becoming more prevalent in weather prediction and meteorological study, as machine learning models can yield high accuracy predictions with very few input parameters, unlike numerical solutions. We saw firsthand here that we only needed one feature to predict the presence of a thunderstorm with over 70% accuracy. For much more complicated problems, it is reasonable to say that machine learning models could be a useful tool in creating fast predictions.

Acknowledgements

The author is thankful to Prof. Natalie Frank and our TA Rohin Gilman for invaluable help throughout the quarter on understanding and implementing the various computational methods discussed throughout the quarter. We are also thankful to our peers for editing and inspiration for this paper.

References

[1] Sevir tutorial. 2024.

[2] R. J. Chase, D. R. Harrison, A. Burke, G. M. Lackmann, and A. McGovern. A machine learning tutorial for operational meteorology. part i: Traditional machine learning. *Weather Forecasting*, 37:1509–1529, 2022.

[3] C. R. Harris, K. J. Millman, S. J. der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, SebastianBerg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020.

[4] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[6] M. Waskom, O. Botvinnik, D. O'Kane, P. Hobson, S. Lukauskas, D. C. Gemperline, T. Augspurger, Y. Halchenko, J. B. Cole, J. Warmenhoven, J. de Ruiter, C. Pye, S. Hoyer, J. Vanderplas, S. Villalba, G. Kunter, E. Quintero, P. Bachant, M. Martin, K. Meyer, A. Miles, Y. Ram, T. Yarkoni, M. L. Williams, C. Evans, C. Fitzgerald, Brian, C. Fonnesbeck, A. Lee, and A. Qalieh. mwaskom/seaborn: v0.8.1 (september 2017), Sept. 2017.